Single-Database Private Information Retrieval Protocols : Overview, Usability and Trends.

Carlos Aguilar Melchor and Philippe Gaborit

***Abstract*— A Private Information Retrieval (PIR) scheme is a protocol in which a user retrieves a record out of N from a replicated database, while hiding from the database which record has been retrieved, as long as the different replicas do not collude.**

A specially interesting sub-field of research, called single-database PIR, deals with the schemes that allow a user to retrieve privately an element of a non-replicated database. In these schemes, user privacy is related to the intractability of a mathematical problem, instead of based on the assumption that different replicas exist and do not collude against their users.

Single-database and replicated-database PIR schemes have generated an enormous amount of research in the privacy protection field during the last two decades. However, many scientists believe, specially for single-database PIR schemes, that these are theoretical tools unusable in almost any situation. It is true that these schemes usually require the database to use an enormous amount of computational power, but considering the huge amount of applications these protocols have, it is important to evaluate precisely their usability.

We present in this article an overview of the current single-database PIR schemes through the innovations they have brought to this field of research. This gives a unified view of the evolution since the first of these schemes was presented by Kushilevitz and Ostrovsky in 1997 and up to the latest trends in single-database PIR research such as trusted hardware usage, and noise-based schemes. Then, we compare the most representative of these schemes with a single set of communication and computational performance measures. We highlight that practical usability of PIR schemes is not as dependent on communication performance as the literature suggests, and that a trade-off between communication and computation leads to much more versatile schemes.

I. Introduction

Usually, to retrieve an element from a database, a user will send a request pointing out which element he wants to obtain, and the database will send back the requested element. Which element a user is interested in may be an information he would like to keep secret, even for the database administrators. For example, the database may be :

- an electronic library, and which books we are interested in may provide information about our politic or religious beliefs, or other details about our personality it may be desirable to keep confidential,
- stock exchange share prices, and the clients may be investors reluctant to divulge which share they are interested in,
- a pharmaceutical database, and some client laboratories may wish that nobody may learn which are the active principles they may want to use,

To protect his privacy, a user accessing a database may therefore want to retrieve an element without revealing which element he is interested in. A trivial solution is for the user to download the entire database and retrieve locally the element he wants to obtain. This is usually unacceptable if the database is too large (for example, an electronic library), quickly obsolete (for example, stock exchange share prices), or confidential (for example, a pharmaceutical database).

Private Information Retrieval (PIR for short) schemes aim to provide the same confidentiality to the user (on which element is requested) than downloading the entire database, with sub-linear communication cost. PIR was introduced by Chor, Goldreich, Kushilevitz, and Sudan in 1995 [1]. In their paper, they proposed a set of schemes to implement PIR through replicated databases, which provide users with information-theoretic security as long as some of the database replicas do not collude against the users.

Remark that PIR schemes do not ensure database confidentiality: a user may retrieve more than a single database element with a PIR scheme without the database learning it. A PIR scheme ensuring that users retrieve a single database element with each query is called a Symmetric PIR (or SPIR) scheme.

In this paper, we will focus on PIR schemes that do not need the database to be replicated, which are usually called single-database PIR schemes. Users' privacy in these schemes is ensured only against computationally-bounded attackers. It is in fact proved that there exists no information-theoretically secure single-database PIR scheme with sub-linear communication cost [1].

A field of research closely related to PIR is the one of Oblivious Transfer [2]. Oblivious Transfer schemes are single-database SPIR schemes except that they aim to limit the computational cost for the user and the database regardless of communication cost. In such schemes, an encrypted version of the whole database is usually sent to the user. Oblivious Transfer is a fundamental cryptographic primitive mostly used in theoretical proofs, the most interesting application being probably that it is complete for secure multi-party computation [3]. As Oblivious Transfer schemes are mainly theoretical tools and answer to different issues than single-database PIR schemes we will not deal with them in this paper.

The first single-database PIR scheme was presented in 1997 by Kushilevitz and Ostrovsky, and since then improved schemes have been proposed by different authors [4], [5], [6], [7], [8], [9].

All of these schemes follow a similar approach, but it is difficult to understand which are the innovations brought by each of them, and the impact that the different innovations have on communication and computational performance. We present in this paper the fundamental approach that all of these schemes follow, and indicate why each of them has meant a step forward.

Single-database PIR schemes are computationally expensive. Indeed, in order to answer a query, the database must process all of its entries. If in a given protocol it does not process some entries, the database will learn that the user is not interested in them. This would reveal to the database partial information on which entry the user is interested in, and therefore is not as

XLIM-DMI, Limoges University, 123, av. Albert Thomas, 87000 Limoges
aguilar@xlim.fr, gaborit@xlim.fr

private as downloading the whole database and retrieving locally the desired entry.

The computational cost for the server is therefore linear on the database size. Moreover, current schemes have a very expensive cost per bit in the database, a multiplication over a large modulus. This limits both the database size and the throughput shared by the users, limiting as well their usage for many databases as for other applications such as low-latency unobservable communications [10] or private keyword search [11]. As a consequence, all of the current single-database PIR schemes are unusable in many applications as it will be shown in section V.

Efforts have been done to reduce the computational cost of the database through the use of trusted hardware [12], [13]. The resulting protocols greatly reduce the computational cost and provide optimal communication cost. However, the computation is done by the trusted hardware, which is much slower than usual hardware. Therefore, even for moderate sizes of the database the computation time remains too large for many applications.

Finally, in a recent work [14] the authors present a noise-based approach. This protocol has not as good communication performance as other single-database PIR schemes, and bases its security on lattices and NTRU-like [15] scrambling assumptions, instead of on number-theoretic intractability reductions. However, it seems difficult to find a scheme that has low computation and communication costs under strong assumptions, despite the interest these protocols awake in the research community. We believe that the approach we presented in [14] is an interesting alternative to number-theory. Thorough study and validation by the research community of noise-based schemes can open the path to single-database PIR schemes usable in a much broader span of applications than the one we can currently reach. Even if this protocol has not been validated by the research community, we present here its performance to show the impact such a scheme can have in PIR applications.

The contribution of this paper is twofold. First, we present a complete survey and concept analysis of the existing single-database PIR schemes. In [16], Gasarch's presents a survey of general PIR schemes, including replicated database schemes, primitives implying and implied by the existence of PIR schemes, theoretic bounds and many other subjects among which single-database PIR is only mentioned but no survey of these schemes is really done. Very recently (in fact in parallel with our work) Ostrovsky and Skeith [**?**] as eprint a transcription of an invited talk to PKC 2007, which surveys single-database PIR schemes but their paper is limited to the description of number theory based PIR schemes and does not consider performances of these schemes. We believe that single-database PIR schemes deserve their own survey for three reasons:

- the number of publications on this subject is large enough,
- this variant is specially interesting as forcing the database to be replicated and supposing that the different replicas do not collude (as it is done in general PIR schemes) is very restrictive,
- the different papers on this research field are often highly technical, have led to re-discoveries, and performance issues are often neglected or evaluated with many different sets of measures.

Indeed, most of the papers on single-database PIR research use many common techniques such as recursion or agglomeration (see section II) mixed with their own innovations and it is often difficult to know what is the real contribution of a given scheme and evaluate its impact on performance. In this survey, we isolate transversal techniques that are usable for any single-database PIR scheme and present each protocol through the innovations it brings to the PIR research field. We also use a unified set of measures to compare from a communication and computation point of view the performances attainable with each of these protocols.

The second major contribution of this paper is to highlight that computational cost is and will probably remain so high in the near future that communication-efficiency in classical schemes is pointless for practical applications. Indeed, the throughput a server can provide to a user is so small when compared with today available bandwidths, even over the Internet, that having a large expansion factor on the communication cost would have a very small impact on bandwidth usage. We show that other, less communication efficient solutions can provide much better trade-offs between communication and computation. Such trade-offs do not appear to be attainable through number-theory, but a noise-based approach seems to bring this possibility.

The paper is organized as follows: in section II we introduce some basic concepts and transversal techniques usable by any PIR scheme. In section III we present an overview of the current single-database PIR schemes. The communication and computational performance analysis of these schemes is done in sections IV and V, and section VI brings the trends analysis. Finally, we conclude in section VII.

II. Basic concepts and common techniques

We describe a database as a set of n l-bit elements. PIR requests are usually formed of a set of n query elements, one per each database element. Each of these query elements is combined with the database element it is associated to, and then the results are combined between them to obtain the PIR reply.

Because of this common approach, some techniques can be used with all the existing PIR schemes. In this section we present first how it is possible to adapt any scheme for any database element size, and second how the recursive usage of PIR schemes leads to much more versatile protocols.

A. Iterative reply generation

It is straightforward to adapt a single-database PIR protocol to any value of l. For example, if a given single-database PIR scheme allows to recover one-bit elements from a database, it can also used to obtain 2-bit elements. When the user sends the PIR request to the database, this one will operate as follows :

- it generates a PIR reply by using the request over the set of n 1-bit elements formed by the first bit of each element in the database,
- it generates a second PIR reply by using the same request over the set of n 1-bit elements formed by the second bit of each element in the database.

Of course, this can be generalized to elements of any size and schemes allowing to retrieve chunks of information of arbitrary size. This is possible because the requests generated by the single-database PIR schemes are always independent from the database contents. When l is larger than the chunk size a scheme allows to retrieve, we will say that the database replies iteratively until the entire l-bit element is sent.

B. Database elements aggregation

When a scheme allows to retrieve chunks of information larger than $\alpha \times l$ bits, α being an integer constant and l being the size of the database elements, a trivial improvement can be done: the database can be seen as composed of n/α elements of size $\alpha \times l$. This does not increase the database reply size and lowers the query size as just n/α query elements are needed instead of n. As this improvement is common to all schemes and is application dependent (as it depends on the database element size), we will not include it on the final performance results, letting the reader evaluate it for her/his own application.

C. Load balancing and recursive usage

If the user sends n query elements and the database a PIR reply, the total communication cost is $O(n)$. To reduce this cost, it is possible to use a load balancing technique which was originally presented in the seminal paper about PIR [1]. The idea is to see the n-element database as a matrix of $\sqrt{n}$ lines and $\sqrt{n}$ columns each scalar in the matrix being a database element. The user sends $\sqrt{n}$ query elements, one for every column in the matrix, and the database replies iteratively sending back $\sqrt{n}$ PIR replies, one for each line in the matrix. As Figure 1 shows, with such an approach, the user retrieves a full column of data, containing the element he is interested in with total communication cost in $O(\sqrt{n})$.

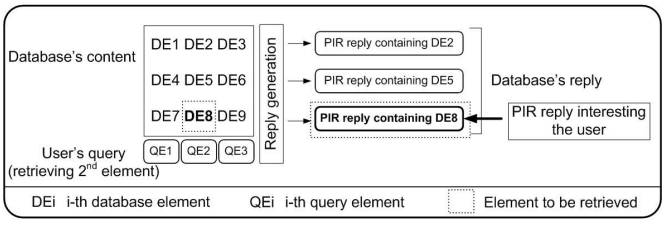

Fig. 1. Load balancing.

When representing the database as a matrix, instead of using the load balancing technique, it is possible to use the PIR scheme recursively as Kushilevitz and Ostrovsky proposed in [17]. The recursive usage of PIR schemes allows to lower the size of PIR requests while increasing the size of the PIR reply in a very versatile way. The main idea is that using the load balancing technique database obtains $\sqrt{n}$ PIR replies and each of them can be seen as an element of a *virtual* database. The user can therefore send a second query to retrieve one of the replies issued from the load balancing process.

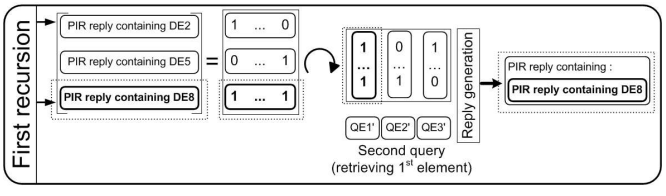

Fig. 2. Recursive usage of a PIR scheme.

In figure 2, we go back to the example given in figure 1 : the first recursion results in three PIR replies that are used as a virtual database for the second recursion. As with the load balancing technique, when recursion is used the query size shrinks (from $O(n)$ to $O(n^{1/2})$), and the reply size increases.

This approach is much more interesting than the load balancing technique for two reasons. The first one was not obvious at the time the scheme was proposed, and comes from the fact that some schemes can implement the recursion in a very efficient way as we will show later. The second reason is that load balancing may be done just once, representing the database as a matrix. Recursion can be done as many times as the number of dimensions that the database representation has.

If the database is represented as a cube of size $n^{1/3}$, the user will send three queries with $n^{1/3}$ elements each. The database will compute a matrix of $n^{1/3} \times n^{1/3}$ PIR replies from the first query. This matrix can be seen as a virtual database and the second query will be used to retrieve one column of $n^{1/3}$ PIR replies. This column will also be used as a virtual database of $n^{1/3}$ elements, and the third query will be used to obtain the PIR reply containing the element the user is interested in. Generally, if the database is represented by a d-dimension hyper-cube, d recursions are possible. With such a representation the user will send d requests, each composed of only $n^{1/d}$ residues. This allows a user to shrink greatly the queries' size, but the size of the PIR replies increases quickly (exponentially in most cases) in d. A trade-off must be made, depending on the application the PIR scheme is used for.

Some PIR schemes [9], [13] do a long pre-computation over the database contents before answering to the PIR queries. Sometimes this pre-computation is mandatory for the scheme to work properly [9], other times it just brings a performance improvement [13]. When using recursion, pre-computation cannot be done over the virtual intermediate databases, as they depend on the users' first queries, and therefore these two techniques are incompatible. Whenever a scheme has a phase of pre-computation, we will thus not consider the usage of recursion.

III. Analysis of the existing single-database PIR schemes

A. Kushilevitz and Ostrovsky's scheme

In [17], Kushilevitz and Ostrovsky created the first single-database PIR scheme, by using quadratic residues. What exactly are quadratic residues is not as important as their properties:

- the user can efficiently generate numbers which are quadratic residues (QRs) and numbers which are quadratic non residues (QNRs),
- the user can efficiently test if a number is a QR or a QNR,
- the user can send sets of such numbers to a database which will be unable to distinguish QRs from QNRs
- there is an operation *OP*, computable by such a database, that from a set of QRs and QNRs gives a QNR if and only if the number of QNRs in the initial set is odd.

This protocol allows a user to retrieve a single bit. We will suppose w.l.o.g. that the database is formed of one-bit elements. If this is not the case, the database will proceed iteratively to send the requested element. The idea behind this PIR scheme is for the query to be constituted of one QR number for each element of the database except for the element to be retrieved and a QNR number for that element. The database computes the operation *OP* over the set of numbers associated with the elements in the database set to one (and ignores the others), and sends the result to the user. If the element the user is interested in is set to one the database will have selected a QNR among the numbers and

the result will be a QNR. If the element the user is interested in is set to zero, the database will have selected only QRs and the result of the operation will be a QR. Figure 3 resumes this idea.

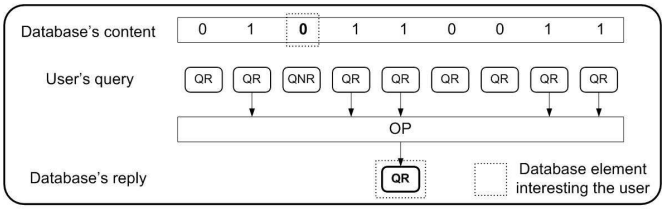

Fig. 3. QRs and PIR schemes.

The first contribution of Kushilevitz and Ostrovsky is therefore an approach to make single-database PIR schemes. In his master thesis [5], Eran Mann formalizes this approach and introduces the notion of homomorphic trapdoor predicates, which are merely the predicates having the properties that we have described at the beginning of this section for quadratic residues. The second major contribution of Kushilevitz and Ostrovsky's paper has already been introduced: the recursive usage of PIR schemes. All of the current single-database PIR schemes follow the approach proposed by Kushilevitz and Ostrovsky. The different schemes use homomorphic trapdoor predicate families with properties that improve significantly the schemes' performance. But, in each of them, the approach is to use, recursively if necessary, the predicates proposed by these authors.

B. Homomorphic encryption based schemes

Apart from the master thesis of Eran Mann, the article by Kushilevitz and Ostrovsky led to another significant work: a paper by Julien P. Stern [4], which made a major outbreak. Stern proposes exactly the same scheme than Kushilevitz and Ostrovsky, except that, instead of using a trapdoor predicate that can only encode one bit of information (for example being a QR or a QNR), he proposes to use homomorphic encryption algorithms, which have all the properties needed, but can encode many bits of information in every number resulting from the OP operation.

In this scheme, instead of sending many QRs and one QNR, the user sends many encryptions of zero and one encryption of one. The protocols can be used with any cryptosystem which has two major properties: indistinguishability and homomorphic encryption. Indistinguishability ensures that only the user generating the cyphertexts can distinguish the numbers that are encryptions[1] of zero and the numbers that are encryptions of one. Homomorphic encryption ensures that the multiplication of two cyphertexts which are encryptions of two cleartexts a and b, results on an encryption of $a + b$ (left side of figure 4).

This multiplication can be iterated into an exponentiation to *absorb* a message as shown in the right side of figure 4. If an encryption of one is raised to the power m it will become an encryption of m whereas if an encryption of zero is raised to the power m it will remain an encryption of zero (as $0 \times m = 0$). This is used by Stern to improve Kushilevitz and Ostrovsky's scheme as the database can encode in a cyphertext as many bits as a

[1] Note that in such cryptosystems, for a given cleartext many different cyphertexts exist.

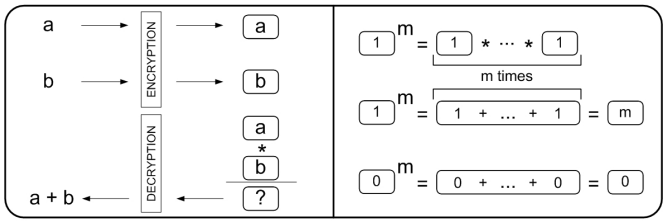

Fig. 4. Homomorphic encryption property.

cleartext can have. Figure 5 shows Stern's scheme. Note that this protocol is exactly the same as Kushilevitz and Ostrovsky's for one-bit elements as if an element is set to zero, the corresponding cyphertext will be ignored (as raising to the zero power results in one, the neutral operand for the multiplication).

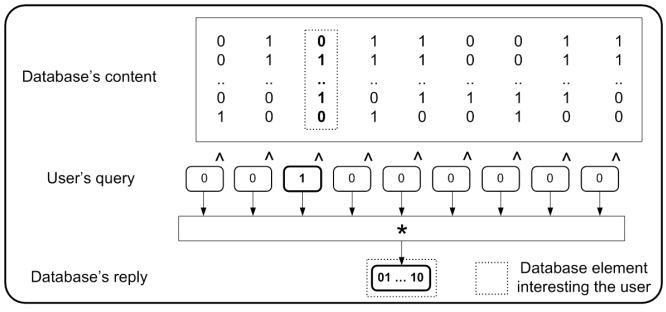

Fig. 5. Stern's scheme.

When users try to retrieve l-bit elements from a database with $l > 1$ this scheme is of course very interesting, since every number sent back can encode many bits contained in the database instead of one. However, even if the user is interested in receiving a single bit of information, the possibility to encode many bits in each number is very interesting. The reason for this is pretty simple : when using the load balancing technique, or recursion, the database must send a number for each bit forming a column (for the load balancing), or for each bit forming a residue (for the recursion). By the usage of homomorphic encryption schemes, the columns and residues can be encoded very efficiently. This is specially important for recursion, as the factor resulting in an exponential growth of the reply size is greatly reduced (nowadays the reduction is of one thousand approximately).

In 2004, there was a rediscovery of Stern's proposal [7], and a proposition by Lipmaa [8] which is basically Stern's construction with the recently discovered length-flexible homomorphic encryption scheme of Damgärd and Jurik [18]. In his paper Lipmaa twists Stern's construction, taking profit of the length-flexible cryptosystem to provide PIR schemes that are both practical and asymptotically interesting. Lipmaa remarks that using correctly this cryptosystem, it is possible to obtain a linear growth in the server reply (instead of exponential) when using recursively the PIR scheme. This greatly improves the versatility of the protocol and leads to an asymptotic behavior much better than with any of the previous schemes.

C. Adaptive predicate schemes

One year after Stern's proposal, Cachin, Micali, and Stadler presented a scheme [6] based on a new trapdoor predicate that they called the ϕ-assumption. This predicate, just as being a

QR or not, can encode just a single bit. Whereas this may seem as a step backwards after Stern's work on the usage of homomorphic encryption schemes, it is not. The main reason is that these trapdoor predicates have a very interesting property : a user can create a compact generator, out of which the database can obtain the numbers forming the query. Query size is thus almost independent of the number of elements in the database (growth is logarithmic), and even if the system is not practically implementable[2], when database size increases this approach beat asymptotically all the PIR schemes published before it.

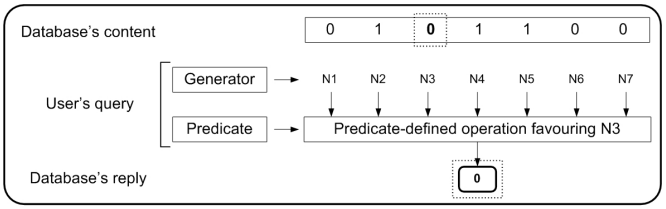

Fig. 6. Cachin Micali and Stadler's scheme.

The basic idea in this scheme is to create first a number generator which will be used locally by the database to obtain a set of numbers, and afterwards create a trapdoor predicate such that the number associated with the index interesting the user has special properties (see figure 6).

This approach in fact has led recently to a very interesting variation. In 2005, Gentry and Ramzan presented a scheme [9], which like Lipmaa's scheme is practical and presents an asymptotical improvement, even if for many applications Lipmaa's construction is better, as shown in the next section. In their paper, the authors present a construction that generalizes the proposal of Cachin et al., and their scheme can be implemented using a slight variation of the ϕ-assumption. Aside from the generalization, two major modifications are done with respect to the initial scheme. The first modification is pretty much the same as Stern did with respect to Kushilevitz and Ostrovsky's scheme: modifying the trapdoor predicate to encode more than a single bit. The second modification is very simple and consists on using the same numbers (which are associated with the database bits) for all the queries. This is almost trivial, but was not proposed by Cachin et al., and it allows to make very small queries that just describe the predicates under which the desired numbers will have special properties.

D. Trusted hardware schemes

The performance attainable with straightforward usage of trusted hardware was analyzed by Smith and Safford in [12]. If the database has a trusted hardware device, such as a secure coprocessor, retrieving an element of the database privately is very simple. The user sends in an encrypted form the index of the element he is interested in to the trusted hardware device. This device reads *locally* the contents of the whole database, and stores only the element the user wants. Finally, the trusted hardware device encrypts the element and sends it back to the user. The cost of such an operation is roughly the one of reading the entire database as trusted hardware I/Os are really slow because of their security constraints.

[2] Queries are too large and the communication rate is too small for almost any application.

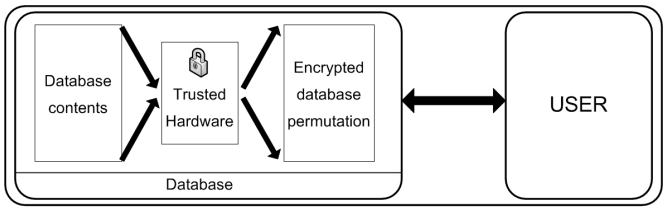

Fig. 7. Pre-computation in Asonov and Freytag's scheme.

An improvement is proposed by Asonov and Freytag in [13]. In this scheme, the trusted hardware device pre-computes an encrypted permutation of the database as shown in figure 7. For the first query sent to the database after the pre-computation, the trusted hardware device just needs to read the encrypted element in which the user is interested (as the database does not know the permutation, it cannot learn which element is being retrieved) and re-encrypt it (so that the user can decrypt the reply) before sending it to the user.

When the database receives the second query after the pre-computation, the trusted hardware device will have to read the encrypted element retrieved for the first query plus another element in order to hide from the database whether the second query is for the same element than the first or not. Generally, to avoid that the database may learn whether the same elements are being retrieved or not on subsequent queries, the trusted hardware must read for each of these queries all the elements already read since the last pre-computation plus one, and of course re-encrypt the requested entry.

The computational cost is therefore roughly linear in the number of queries since the last pre-computation, instead of linear on the database size. With this scheme a server will be able to answer much quicker than with any other scheme to some PIR requests before having to do a pre-computation again. However, the pre-computation is very costly and therefore this scheme is mostly adapted to small databases that change very scarcely (as the pre-computation must be done every time the database changes), receiving a small amount of queries.

A major issue with this approach is that user's trust must be placed on hardware tamper resistance but also on the entity programming the device. This device cannot therefore be programmed by the server administrators, and a trusted third party is required. Number-theory and noise-based approaches require no trusted entities to step in at any time.

E. Noise-based schemes

Very recently, the authors proposed in [14] a new scheme which difficulty is based on lattice scrambling, the same kind of problems used for the NTRU scheme. In this protocol the user has a local matrix generator from which it is possible to obtain sets of matrices belonging to a secret lattice[3]. These matrices are disturbed by the user by the introduction of noise in two thirds of the matrices' columns as shown in figure 8 to obtain respectively softly disturbed matrices (SDMs) and hardly disturbed matrices (HDMs).

To obtain an element from the database the user sends a set of SDMs and one HDM (replacing respectively the QRs and QNR

[3] A lattice is a vector space like algebraic structure.

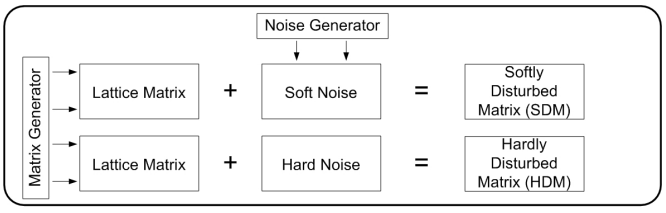

Fig. 8. Matrix perturbation.

in Kushilevitz and Ostrovsky's scheme). The database inserts each element in the corresponding matrix with a multiplicative operation OP and sums all the rows of the resulting matrices to obtain the database reply, a single noisy vector (see figure 9). Using the unmodified columns of the matrices sent in the request, the user is able to find the noise associated to the returned noisy vector. If the soft noise multiplied by the total noise factor (which is proportional to the number of elements in the database) is much smaller than the hard noise, it can be filtered out and the user can retrieve the information associated to the noise of the HDM matrix.

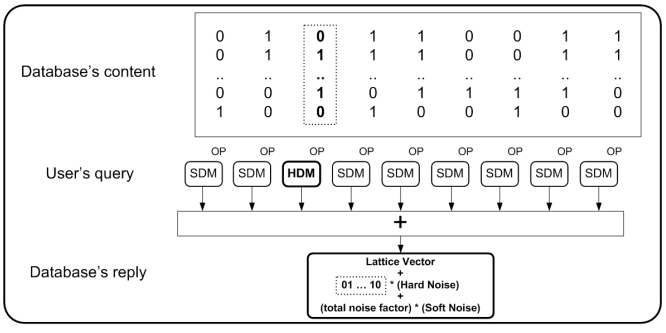

Fig. 9. Noise-based approach.

This scheme has a communication performance not as good as other schemes presented in this section, but all the operations are multiplicative (instead of exponential) or additive (instead of multiplicative) and no trusted hardware is used. As it will be shown in the performance analysis, the computational cost reduction is very significant, and in many cases reducing this cost is imperative for the usability of a PIR protocol.

IV. Communication performance

When querying a database without trying to hide which element is being retrieved, a user sends the index of the entry he is interested in and the database sends back the requested element. Roughly, if the database has n elements, the query size will be $log(n)$ bits, and the reply expansion factor will be 1.

When using a PIR scheme, the reply expansion factor may depend on the size of the element retrieved as information is sent by *chunks*. Usually, a PIR reply has a given size and can encode a fixed number of bits. If the database elements are smaller than the chunk size the expansion factor will grow. For the sake of readability we will not consider this case. To compare the PIR schemes between them we will just present query size and database reply expansion factor, supposing that database elements are large enough. Trusted hardware schemes always have optimal communication cost (query size is $log(n)$ and reply expansion factor is 1) and therefore they are not included in this comparison.

In table 10, the use of the recursive construction proposed by Kushilevitz and Ostrovsky is represented by a parameter noted d, $d = 1$ meaning that no recursion is done. The integer k represents a factorization-type security parameter, which for practical applications should not be lower than 1024. We have taken a different notation for Cachin et al.'s security parameter, K, as the authors fixed $K > log^2(n)$, even if their only non-factorizable integer had K^5 bits. This gives a much stronger constraint than $k > log^3(n)$, i.e., the usual asymptotic estimation against factorization. The reason for this is that the constraint is introduced to ensure the security of their number generator out of which their query is formed by the database. As the security assumptions done are different, we use different notations for the security parameters. Finally, s represents an integer parameter that can be fixed by the user. As it will be shown in section IV, the computational cost of the schemes having this parameter is at least in $O((sk)^2)$ per bit in the database and therefore s must be kept close to one, in order to limit the computational cost. For Aguilar and Gaborit's scheme (noted hereafter AG), the parameter N represents a security parameter for the lattice used, typically $N = 66$, the parameter l_0 represents the size in bits of the soft error magnitude, it has a logarithmic relation with n, typically $l_0 = 20$.

The performance results presented in figure 10 for the scheme proposed by Kushilevitz and Ostrovsky (KO) are not exactly the same as the ones presented in their paper. These authors stayed with some load balancing, instead of pushing the recursive scheme to its maximum level. This strategy has been abandoned on current schemes and therefore to give a better comparison we have provided the results for the maximum recursive scheme rather than for the scheme with load balancing. Stern's and Lipmaa's schemes can be implemented with various encryption algorithms. The results presented in the table represent an implementation with the Damgärd-Jurik cryptosystem, which is the most effective and versatile homomorphic cryptosystem to date. Using Gentry and Ramzan's scheme (GR) recursively is not studied as this scheme uses pre-computation (see end of section II-C). The dimension d is therefore not indicated in the results associated to this scheme. For Cachin et al. (CMS), we have not included d in the performance results, since it was designed as a theoretic scheme. The reason it has been included in figure 10 is to make visible the impact on asymptotic performance resulting from the innovations they introduced.

Indeed, the results illustrate the impact of the innovations presented in the previous section. The drastic reduction of the database reply expansion factor (specially when $d > 1$) that can be observed between the first and the second lines is the result of Stern's introduction of chunk sizes greater than unity. The dependence on n of query sizes in lines four and five is just reduced to the asymptotic behavior expected for K, which is the result of Cachin et al.'s approach of generating the trapdoor predicates *after* defining the numbers forming the queries. Lipmaa's usage of length-flexible cryptosystems lowers from geometric to linear the increase of Stern's replies as d grows. Finally, the possibility of retrieving more than one bit per reply, and the replacement of Cachin et al.'s number generators by a fixed set of numbers by Gentry and Ramzan, gives the first PIR scheme with a communication cost independent of n and efficient

in practice[4].

Numerical values in figure 10 show that Gentry and Ramzan's scheme is the most communication efficient protocol. Queries are very small and the reply expansion factor is only 4. However, if database elements are large, Stern's and Lipmaa's should be used as they have the lowest reply expansion factor. On the opposite side, the noise-based approach is clearly less efficient from a communication point of view, as either queries are really large (396Mb for $d = 1$) or the database reply expansion factor becomes much larger than the ones obtained with the other protocols (20 for $d = 2$).

V. Computational performance

The exact evaluation of computational complexity in PIR schemes leads to difficult to analyze results. For this reason, we give in this section simple lower bounds on this complexity to represent the magnitude of the computational costs, accepting a lost in accuracy for the sake of readability. Reply generation will be the limiting factor for single-database PIR usage, we will therefore not analyze the computational cost for the users to generate a query and to decode a PIR reply.

Lipmaa's and Stern's schemes are clear improvements over the scheme proposed by Kushilevitz and Ostrovsky and therefore we will not present in this section the computational cost associated to this scheme. For the same reason we will just present the computational cost for the protocol proposed by Gentry and Ramzan and not for the one initially proposed by Cachin et al. For all of these schemes, reply generation cost is roughly a modular multiplication per bit on the database.

Schemes based on trusted hardware require much less computation. In the scheme proposed by Smith and Safford (noted SS hereafter), the cost of generating a reply is roughly the one of reading the entire database for the trusted hardware device. With Asonov and Freytag's scheme (AF), the base computational cost is reading an element of the database for the trusted hardware device and it grows linearly in the number of queries since the last pre-computation.

Figure 11 presents the computational costs of these schemes in bit operations. All of the measures are given for l-bit database elements. Pre-computation is done for a given database content. If an element of the database changes, the computational cost to update the database pre-computation is given by the *update on write* measure. If this cost is given for a protocol it must be multiplied by the number of elements that change, if no cost is given it means that the pre-computation must be completely done again every time any element changes in the database.

We have noted ***M(k)*** the cost of a k-bit modular multiplication, and ***THR(l)*** the cost of reading l bits with a trusted hardware device. The parameter α represents the number of queries treated since the last pre-computation in Asonov and Freitag's scheme. As previously, k is a security parameter representing the length of a hard-to-factor modulus, d the dimension of the database representation, n the number of elements on the database, and s an integer parameter of Damgärd and Jurik's homomorphic encryption scheme.

[4]In an asymptotic evaluation we must suppose $k > O(log^3(n))$ [9]. Furthermore this scheme is based on the existence of enough prime numbers lower than $2^{k/5}$, however for any practical parameters there is no need to increase k above the factorization limit we have fixed. Indeed, if $k = 1024$, the results presented here are valid for $n < 10^{58}$.

Figure 12 presents the evaluation of these performance results. When Stern's or Lipmaa's schemes are used the database reply generation computational cost is a modular multiplication per bit in the database. The size of the modulus will be 2048 bits for current factorization standards and $s = 1$. With Gentry and Ramzan's scheme the cost will be a modular multiplication over 1365 bits per bit in the database. Finally, when Aguilar and Gaborit's scheme is used, the database must compute $\frac{3}{2}N \simeq 100$ additions of $3l_0 = 60$ bits per bit in the database. Our Opteron 248 server can compute two hundred thousands 2048-bit modular multiplications, four hundred thousands 1365-bit modular multiplications, and four billion 60 bit additions per second. For the schemes based on trusted hardware, we will use the performance results from [19] in which an IBM 4758 secure co-processor is used. Last generation trusted hardware devices have a USB2 communication interface, however, due to obfuscation needs, the I/O throughput is usually very low. In the case of the IBM 4758 this value is 1 Mbyte per second.

When comparing the closed formulas of figure 12, the first important remark is that using trusted hardware without pre-computation only improves the bandwidth a server can provide by one order of magnitude when compared to number theory schemes. On the other side, pre-computation in Asonov and Freitag's scheme allows a server to provide rapid replies to PIR requests as long as the number of queries done between each pre-computation is much smaller than n. Pre-computation is theoretically in $O(n\sqrt{n})$, but the authors of this scheme assert that experimentally it is closer to $O(n)$. We have kept this optimistic approach for the results even if it should be tested thoroughly before acceptance. For a database of one thousand songs of 2 Mbytes each, the trusted hardware device needs roughly a whole day of pre-computation. Pre-computation is therefore very expensive, and this approach should only be used for small non-volatile databases (to limit pre-computation time) with a restricted number of queries per day (when $\alpha \simeq n$ the pre-computation benefit is lost). With Gentry and Ramzan's scheme, pre-computation cost is roughly similar (14 hours), however two major differences must be considered. First, updating the pre-computation has a very moderate cost for medium size databases (50 seconds per element changed). Second, the pre-computation remains valid independently of the number of queries received. This scheme's pre-computation is therefore much less restrictive than with Asonov and Fretag's scheme. On the other side, pre-computation for large databases is too costly. If the database had one hundred thousand elements of 2Mb, pre-computation would need 16 years (!).

The bandwidth attainable with the noise-based approach is one order of magnitude larger than with Smith and Safford's scheme and between two and three times larger than with number theory based schemes. However, it is important to keep in mind that query sizes are large and therefore this scheme should not be used to retrieve small elements from a database as the cost of sending the query would not be amortized.

Numerical values in figure 12 show that the computational cost is so large that, even for a medium size database, the throughput a server can deliver with a PIR scheme using all of his computational power is very limited. In the given example the private retrieval will use at most 0.016 percent of a 10Mb ADSL connection. Even with a database reply expansion factor of 1000 the bandwidth usage with a number-theory scheme would only

be of 4 percent. Database reply expansion factor is therefore a minor parameter and users should be careful not wasting other resources to limit it.

Considering these results we can remark that Gentry and Ramzan's scheme is always a better choice than Stern's or Lipmaa's schemes. Likewise, using the noise-based approach with $d = 2$ will only induce a bandwidth usage of 8 percent. Considering that with such an approach the user will download the database elements at 40Kbits/s instead of 400bits/s (with Gentry and Ramzan's scheme), it comes out that the noise-based approach leads to a much more usable scheme than the other communication optimized approaches.

VI. Usability trends

In this section we examine the usability of PIR, related to the evolution of computational power and network bandwidth. These two values follow respectively Moore's law, which roughly stands that computational power doubles every 18 months, and Nielsen's law, which stands that network bandwidth grows at least by 50% every year [20].

Whether single-database PIR schemes are or will become usable in real life applications is studied in [21]. In this paper the authors compare the time needed to treat a database bit with the time to send it. They remark that if treating a bit takes longer than sending it, it is faster to send the whole database to the user than to compute a PIR reply. They notice that nowadays sending the database to a user with an Internet home connection is ten times faster than computing a PIR reply with number theory schemes. Moreover they prove that if Nielsen's and Moore's laws are respected this will remain so in the future. Remark that the presented noise-based scheme being one hundred times faster than number theory schemes, this situation is inverted, and as the results of this section will show it will remain so. However, we believe that this measure is questionable for various reasons.

First, the database bandwidth used must be paid every month while investing in computing power to lower the communication costs may be done only once. The question is therefore not just which is the faster but also which is the cheaper for the database server.

Second, available bandwidth is a difficult to control factor. Of course, if a server rental approach is used increasing the database bandwidth is not a problem. On the other side, common users have a maximum bandwidth which is fixed by their provider's technology and it is often not possible or very costly to multiply the connection speed over this maximum. If server rental is not possible and the database server is inside a company changing the company's network to upgrade its bandwidth can be a very costly operation just as requesting the installation of landlines to scale up the Internet bandwidth. In some cases, it is just impossible to increase the bandwidth, for example if satellite communications are used, or if situated in an isolated place etc. Computational power on the other hand is very easy to increase, specially as PIR schemes are easy to distribute.

We therefore prefer studying the evolution of the obtainable throughput for the different schemes, and compare the results between them. We discuss the absolute values as well as the relative ones when compared to the evolution of the available bandwidth. Finally, we study the evolution of bandwidth usage by the PIR schemes to try to infer the importance of communication efficiency in the future.

We saw that there were mainly three kinds of PIR schemes: number theory based schemes, hardware schemes and the recent noise-based schemes. To analyze the trends of the different approaches, we have chosen the most versatile schemes of each approach: Gentry and Ramzan's for number theory, Smith and Safford's for trusted hardware, and Aguilar and Gaborit's as an example of what can be achieved with a noise-based approach.

The security parameter of a number theory scheme is the size of an RSA modulus (or discrete log). The size officially advised by international organism such as the NIST was 768 bits in 1995, 1024 bits until 2010, 2048 bits until 2030 and 3072 bits beyond [22].

For trusted hardware schemes the situation is different since no law predicts the evolution of their I/O bandwidth, which is the major parameter for PIR performance. However, as obfuscation through cryptographic operations is the origin of the limited bandwidth in these devices, it is reasonable to suppose that the relation between computational power and I/O bandwidth is linear. We will therefore use Moore's law to predict the evolution of PIR performance with trusted hardware.

This prediction is less reliable than the ones realized for number theory or noise-based schemes and the results presented in this section for trusted hardware are merely indicative. In any case, this approach is very different from the other two. First, because this hardware is costly and usually not installed on servers, and second because a trusted third party is needed to install the software on it. Still, we believe that comparing the other approaches' performances to trusted hardware schemes is interesting and therefore their performance prediction, even if less reliable, has been included in this section.

The scheme we have presented to illustrate the performance attainable with a noise-based approach has a security parameter N related to the size of the matrices forming the PIR queries. The presented results for communication and computational performance correspond to a security parameter of $N = 66$, which should increase in time. To preserve the same computational complexity than the one obtained by NIST recommendations on factorization we will set $N = 66$ until 2010, $N = 80$ until 2020 and $N = 100$ for 2030 and beyond.

Year		2007	2010	2020	2030	2040
Available bandwidth (bits/s)		10M	34M	1.9G	110G	6.3T
Throughput (bits/s)	Number Theory	400	400	40K	1.8M	180M
	Trusted Hardware	8K	32K	3.2M	320M	32G
	Noise-Based	40K	120K	12M	1G	100G

Fig. 13. Throughput trends.

In figure 13, we give the evolution in time for $n = 1000$ of the throughput that a user can obtain when retrieving privately an element from a database. These results are linear in $1/n$ and therefore it is easy to deduce the throughput for any other value of n. The figure takes into account Moore's law, Nielsen's law and size evolution of the security parameters.

Notice that noise-based schemes will improve their speed in comparison to number theory schemes since the size of their key increases slower (linear instead of quadratic). Trusted hardware schemes' speed increases even faster as we haven't taken into account the increase in obfuscation complexity on time. The figure's results show that the span of applications in which PIR

protocols are usable will become larger and larger as throughput will be able to handle large amounts of voice (2010s), high quality audio (2020s) and video (2030s).

For databases with millions of elements, the throughput results given in figure 13 are divided by one thousand. Similarly, if simultaneous PIR queries are authorized, this throughput is to be shared by all the users downloading from the database. This must be taken into account when considering this section results. For example, nowadays, a single user downloading privately a 2Mb song from a one thousand songs database would retrieve it in 5 minutes roughly, using the presented noise-based scheme. However, downloading in a few minutes a song from a one million songs database in which one thousand users are sharing the server throughput does not seem feasible with any existing protocol before 2040.

The presented noise-based scheme's throughput is between 100 and 1000 times larger than number theory schemes. The main weakness of this scheme is the large size of the query ($25Mb$ for $d = 2$ and $n = 1000$). With a current 10Mb download / 1Mb upload connection, query time remains tolerable for medium database sizes (25s for the given parameters), but is unacceptable for large databases in many applications. For example, if $n = 10^6$, query size will be roughly 792Mb for $d = 2$, which implies that the user will have to wait thirteen minutes (while the query is being sent) before the download begins. However, FTTH is already broadly used in Japan, and deployments are scheduled for the coming years in Europe and North America. It seems therefore reasonable to think that symmetric connections will generalize and that query times will be very low even for very large databases in the coming years.

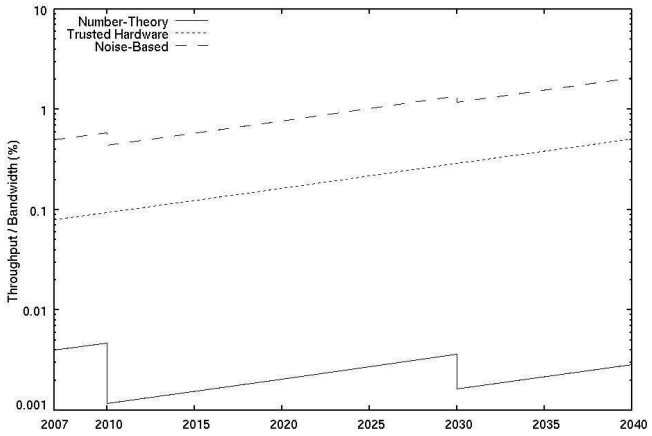

Fig. 14. Relative throughput trends.

If PIR applications will probably rise, general use of these protocols does not seem probable. Indeed, as shown in figure 14, the obtainable throughput for a PIR scheme will remain a very small fraction of the users' bandwidth and therefore non-private retrieval will remain much faster.

On the other hand, PIR schemes are highly parallelizable. Using clusters of low-cost specialized hardware may rise the obtainable throughput to values closer to non-private retrieval performances at a moderated prize. Given the results of figure 14 for the noise-based scheme, a card containing one hundred low-cost specialized units providing each the same throughput than a common processor would suffice. For such a scheme, as operations are additive, each unit would just have to contain simple multipliers and mostly multiplexers. Number-theory schemes, would need ten thousand specialized units providing each the same throughput than a common processor, and each of these units would have to be based on Montgomery multipliers and other costly hardware, and therefore will lead to thousand of times more costly solutions. This approach does neither seem possible with trusted hardware given the high cost of these devices.

Finally, we consider the evolution of the trade-off between communication and computation. If we multiply the results of figure 14 by the expansion factors of the schemes used we obtain the percent of bandwidth that the PIR protocols will need. Trusted hardware and number theory schemes remain almost unchanged and the noise-based scheme rises to an interval between eight percent (nowadays) and forty percent (in 2040). It is important to remark that this approach will remain the only one for which there will be a reasonable trade-off between communication and computation (that results in a throughput between 100 and 1000 times larger). Low communication expansion factors will probably go on being useless as well for trusted hardware as for number theory schemes as bandwidth usage will remain insignificant for these approaches as shown in figure 14.

As a conclusion for this section we may say that time passing by PIR will become usable for more and more applications. However, if computational power and available bandwidth go on following Nielsen's and Moore's laws, retrieving privately an element from a database will be much slower than a non-private retrieval, even for moderated size databases. Specialized hardware may limit this gap, specially for an additive approach such as the noise-based scheme we have presented. Whether specialized hardware is used or not, the performance evolution shows that a multiplicative (number theory) approach will severely limit the applications on which PIR schemes are usable. Trusted hardware, if not specially designed for this purpose does not seem to be a solution either. Finding an additive solution, like the noise-based approach, seems to be the best way to obtain one day good enough performances for PIR usage generalization.

VII. Conclusion and future work

In this paper, after giving a precise analysis of the main ideas used in the existing Single Database PIR schemes and techniques, we have proposed a set of benchmarks of the different PIR schemes in order to compare their efficiency. We have presented three main approaches to obtain PIR schemes: number theory, noise-based, and trusted hardware.

The number theory approach provides secure schemes and relies on well known problems, but is very slow in practice and results in difficult to use schemes for real life applications. The trusted hardware schemes are faster, but the used devices are costly, and a trusted third party is needed to program these devices, which is incompatible with many applications. Eventually, the noise-based approach provides an even faster scheme which, on the other hand, needs the user to send large queries and whose security relies on less studied problems.

The main advantage of the noise-based schemes is to be structurally additive in comparison to number theory based protocols which are multiplicative. Indeed, the performance analysis done in this paper highlights that the multiplicative structure of number theory schemes leads to throughputs that are and will remain very low. The results we have presented show that a practical

development of PIR schemes is related to the existence of faster schemes like the one presented to illustrate the noise-based approach, and more generally to the existence of structurally additive schemes. Moreover, the drawbacks of this approach should step down in time: queries should be sent in at most some seconds for almost any database size as bandwidth growths, and database reply expansion factor's impact on bandwidth usage will remain limited. The issues with number theory (multiplicative schemes) and trusted hardware (cost and trust) are, on the other hand, inherent to these approaches and will probably not fade in time.

Although noise-based protocols are rather new for PIR, they represent a very interesting alternative to number theory based protocols, in the same way that lattice based protocols like NTRU represents an alternative to number theory based cryptosystems. In fact, they seem to be the only potential practical solution to obtain a throughput close to the one of a non-private retrieval. Meanwhile, like for the NTRU cryptosystem, their security reduction to well studied problems is not as straightforward as for number theory based protocols.

To conclude, we ask the following questions: is it possible to construct (additive) noise-based PIR protocols faster than the one presented or which have a smaller query size ? Are there other additive approaches to be explored to obtain rapid PIR schemes ? We hope that this paper will motivate research in these fields to obtain the performance improvement needed by PIR schemes to be considered by the community as the real practical and useful tools they can be instead of theoretical primitives.

REFERENCES

[1] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private Information Retrieval," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995, pp. 41–50.

[2] "All-or-Nothing Disclosure of Secrets", author = "Gilles Brassard and Claude Crépeau and Jean-Marc Robert," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 263. Springer, 1986, pp. 234–238.

[3] J. Kilian, "Founding Cryptography on Oblivious Transfer," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, 2–4 May 1988, pp. 20–31.

[4] J. P. Stern, "A New Efficient All-Or-Nothing Disclosure of Secrets Protocol." in *ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 1514. Springer, 1998, pp. 357–371.

[5] E. Mann, "Private Access to Distributed Information, Technion Master's Thesis, Israel," 2004.

[6] C. Cachin, S. Micali, and M. Stadler, "Computationally Private Information Retrieval with Polylogarithmic Communication," in *Advances in Cryptology: Proceedings of EUROCRYPT*, 1999, pp. 402–414.

[7] Y.-C. Chang, "Single Database Private Information Retrieval with Logarithmic Communication," in *ACISP: Information Security and Privacy: Australasian Conference*, 2004, pp. 50–61.

[8] H. Lipmaa, "An Oblivious Transfer Protocol with Log-Squared Communication." in *ISC*, ser. Lecture Notes in Computer Science, vol. 3650. Springer, 2005, pp. 314–328.

[9] C. Gentry and Z. Ramzan, "Single-Database Private Information Retrieval with Constant Communication Rate," in *ICALP: Annual International Colloquium on Automata, Languages and Programming*, 2005, pp. 803–815.

[10] C. Aguilar Melchor and Y. Deswarte, "From DC-nets to pMIXes: multiple variants for anonymous communications," in *Fifth IEEE International Symposium on Network Computing and Applications*, 2006, pp. 163–172.

[11] R. Ostrovsky and W. E. Skeith III, "Private Searching on Streaming Data," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 3621. Springer, 2005, pp. 223–240.

[12] S. W. Smith and D. Safford, "Practical server privacy with secure coprocessors," *IBM Systems Journal*, vol. 40, no. 3, pp. 683–695, 2001.

[13] D. Asonov and J. C. Freytag, "Almost optimal private information retrieval," in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, vol. 2482. Springer, 2002, pp. 209–223.

[14] C. Aguilar Melchor and P. Gaborit, "A Lattice-Based Computationally-Efficient PIR Protocol,," Under submission, 2006, `http://www.unilim.fr/pages_perso/carlos.aguilar/pir.zip`.

[15] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A Ring-Based Public Key Cryptosystem," in *3rd International Algorithmic Number Theory Symposium (ANTS)*, ser. Lecture Notes in Computer Science, vol. 1423. Springer, 1998, pp. 267–288.

[16] W. Gasarch, "A Survey on Private Information Retrieval", journal = "Bulletin of the European Association for Theoretical Computer Science," vol. 82, pp. 72–107, Feb. 2004.

[17] E. Kushilevitz and R. Ostrovsky, "Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval (extended abstract)," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1997, pp. 364–373.

[18] I. Damgård and M. Jurik, "A Length-Flexible Threshold Cryptosystem with Applications," in *ACISP 2003*, 2003, pp. 350–364.

[19] D. Asonov, *Querying Databases Privately: A New Approach to Private Information Retrieval*, ser. Lecture Notes in Computer Science. Springer, 2004, vol. 3128.

[20] "J. Nielsen , Nielsen Law of internet bandwidth,," `http://www.useit.com/alertbox/980405.html`, 1998.

[21] R. Sion and B. Carbunar, "On the Computational Practicality of Private Information Retrieval," in *14th Network and Distributed Systems Security Simposium, San Diego, CA, 28th February - 2nd March*, 2007.

[22] "National Institute of Standard Technologies (NIST), The key management guideline,," `http://csrc.nist.gov/CryptoToolkit/tkkeymgmt.html`, 2005.

Approach	Scheme	Query size			Reply expansion factor		
		Closed formula	n = 1000		Closed formula		
			d = 1	d = 2		d = 1	d = 2
Homomorphic Trapdoors	KO	$d \times k \times n^{\frac{1}{d}}$	1Mb	63Kb	k^d	1K	1M
	Stern	$d \times (s+1) \times k \times n^{\frac{1}{d}}$	2Mb	126Kb	$\left(\frac{s+1}{s}\right)^d$	2	4
	Lipmaa	$d \times \left(s + \frac{d+1}{2}\right) \times k \times \left(n^{\frac{1}{d}} - 1\right)$	2Mb	126Kb	$\frac{(s+d)}{s}$	2	3
Adaptive Predicates	CMS	$K^4 + 2 \times K^5$	N/A		K^5	N/A	
	GR	$8/3 \times k$	3Kb		4		
Noise-Based	AG	$d \times 3l_0 \times \frac{3}{2}N^2 \times n^{1/d}$	396Mb	25Mb	4.5^d	4.5	20

Fig. 10. Communication performance comparison.

Approach	Scheme	Pre-computation	Update on write	Reply generation
Number Theory	Stern&Lipmaa	—	—	$l \times n \times \boldsymbol{M}((s+1) \times k)$
	GR	$2 \times l \times \frac{k}{3} \times n^2$	$2 \times l \times \frac{k}{3} \times n$	$l \times n \times \boldsymbol{M}(4/3 \times k)$
Trusted Hardware	SS	—	—	$\boldsymbol{THR}(l \times n)$
	AF	$\boldsymbol{THR}(l \times n^{1.5})$	—	$\boldsymbol{THR}(l \times (\alpha+1))$
Noise-based	AG	—	—	$l \times n \times 3l_0 \times \frac{3}{2}N$

Fig. 11. Computation performance comparison.

Approach	Scheme	Pre-computation (sec)		Update on write (sec)		Reply generation (bits/s)	
		Closed formula	n = 1000 l = 2MB	Closed formula	n = 1000 l = 2MB	Closed formula	n = 1000
Number Theory	Stern&Lipmaa	—	—	—	—	$\frac{2\times10^5}{n}$	200
	GR	$\frac{l\times n^2}{3\times10^8}$	50K	$\frac{l\times n}{3\times10^8}$	50	$\frac{4\times10^5}{n}$	400
Trusted Hardware	SS	—	—	—	—	$\frac{8\times10^6}{n}$	8K
	AF	$\frac{l\times n}{2\times10^5}$	80K	—	—	$\frac{8\times10^6}{(\alpha+1)}$	$\frac{8M}{(\alpha+1)}$
Noise-based	AG	—	—	—	—	$\frac{4\times10^7}{n}$	40K

Fig. 12. Computation performance comparison.